(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－184717

(43)公開日　平成11年(1999)７月９日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/46	３４０	Ｇ０６Ｆ　9/46	３４０Ｅ

審査請求　未請求　請求項の数７　ＯＬ　（全 12 頁）

(21)出願番号　特願平9－353092

(22)出願日　平成９年(1997)12月22日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　藤原　靖
神奈川県川崎市幸区柳町70番地　株式会社東芝柳町工場内

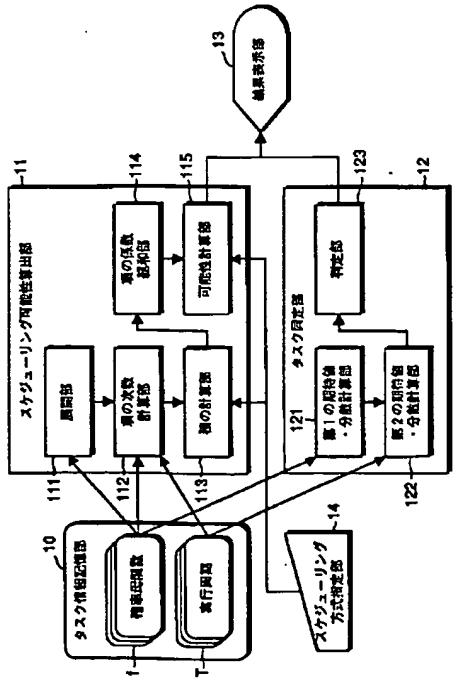
(74)代理人　弁理士　木内　光春

(54)【発明の名称】　スケジューリング解析装置及び方法並びにスケジューリング解析用ソフトウェアを記録した記録媒体

(57)【要約】

【課題】　適用範囲が広く、タスクの実行時間の期待値に基づいて情報を提示するスケジューリング解析の技術を提供する。

【解決手段】　スケジューリング可能性算出部１１の展開部１１１は、有理式として与えられた確率母関数ｆを多項式に展開する。項の次数計算部１１２が、各タスクの確率分布関数に現れる各項の次数を、当該タスクの実行周期の逆数倍する。積の計算部１１３が、項の次数が逆数倍された各タスクの多項式同士を掛け合わせることによって積を計算する。項の係数総和部１１４が、掛け合わされた前記積に現れる各項のうち、次数がスケジューリング方式に対応する基準値より大きい項は廃棄し、次数が基準値以下の項のみを選んで項の係数の総和を計算する。最後に、可能性計算部１１５が、計算された係数の総和に基づいてスケジューリング可能性を計算して結果表示部１３に出力する。



1

【特許請求の範囲】
【請求項１】　スケジューリング方式を指定する手段と、
実行すべき複数のタスクの実行周期又はデッドラインを指定する手段と、
前記タスクの実行時間の確率的分布を用いて、指定されたスケジューリング方式において前記周期内又はデッドラインまでに全てのタスクが終了する可能性を示す値を算出する手段と、
を備えたことを特徴とするスケジューリング解析装置。
【請求項２】　前記算出する手段は、
スケジューリングの対象とする各タスクごとに、多項式又は有理式として表された実行時間の確率分布関数、及び実行周期が与えられた場合に、
有理式として与えられた確率分布関数を多項式に展開する手段と、
各タスクの多項式に含まれる項の次数を、当該タスクの実行周期の逆数倍することによって、各タスク毎の第２の多項式を作成する手段と、
各タスク毎の第２の多項式同士の積を計算することによって第３の多項式を作成する手段と、
前記第３の多項式から、次数が、与えられたスケジューリング方式に応じて予め定めた基準値以下の項の係数の和を計算する手段と、
前記係数の和に基づいて、前記スケジューリング方式の下で、与えられたデッドラインまでに全てのタスクが実行を終了する可能性を計算する手段と、
を備えたことを特徴とする請求項１記載のスケジューリング解析装置。
【請求項３】　スケジューリングの対象とする各タスクに関する情報に基づいて、処理のデッドラインを破る原因となる可能性が高いタスクを同定する手段を備えたことを特徴とする請求項１又は２記載のスケジューリング解析装置。
【請求項４】　前記同定する手段は、
スケジューリングの対象とする各タスクの実行時間の確率分布関数に基づいて、１回の実行あたりの実行時間について、期待値及びその分散を計算する手段と、
１回の実行あたりの前記期待値及びその分散と当該タスクの実行周期とに基づいて、タスク毎のＣＰＵ使用率の期待値及びその分散を計算する手段と、
タスク毎に計算されたＣＰＵ使用率の期待値又はその分散のうち少なくとも一方に基づいて、デッドラインを破る原因となる可能性が高いタスクを判定する手段と、
を備えたことを特徴とする請求項３記載のスケジューリング解析装置。
【請求項５】　スケジューリング方式を指定するステップと、
実行すべき複数のタスクの実行周期又はデッドラインを指定するステップと、

2

前記タスクの実行時間の確率的分布を用いて、指定されたスケジューリング方式において前記周期内又はデッドラインまでに全てのタスクが終了する可能性を示す値を算出するステップと、
を含むことを特徴とするスケジューリング解析方法。
【請求項６】　スケジューリングの対象とする各タスクに関する情報に基づいて、処理のデッドラインを破る原因となる可能性が高いタスクを同定するステップを含むことを特徴とする請求項５記載のスケジューリング解析方法。
【請求項７】　コンピュータを用いて複数のタスクのリアルタイムスケジューリングに関する解析を行なうスケジューリング解析用ソフトウェアを記録した記録媒体において、
当該スケジューリング解析用ソフトウェアはコンピュータに、
スケジューリング方式の指定を受け付けさせ、
実行すべき複数のタスクについて実行周期又はデッドラインの指定を受け付けさせ、
前記タスクの実行時間の確率的分布を用いて、指定されたスケジューリング方式において前記周期内又はデッドラインまでに全てのタスクが終了する可能性を示す値を算出させることを特徴とするスケジューリング解析用ソフトウェアを記録した記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、リアルタイムスケジューリングに関するもので、より具体的には、与えられたタスク群を指定されたスケジューリング方式でスケジューリングした場合に、各タスクの実行がデッドラインまでに終了する可能性を算出する技術を提供するものである。
【０００２】
【従来の技術】コンピュータとそのプログラムとを組み込んだ電気機器を、組み込みシステムと呼ぶ。組み込みシステムのうちには、時間的な制約が厳しいリアルタイム処理の用途に用いられるものが多く存在する。リアルタイム処理では、各タスクは、与えられたデッドラインすなわち制限時刻までに一定の処理を終えなければならないという制約があり、このような制約をリアルタイム制約と呼ぶ。このリアルタイム制約の遵守が強く求められるシステムとしては、航空機管制のような軍事・公共システムなどが典型的であるが、最近では、自動車の燃料噴射制御システムやマルチメディアシステムなどのように、民生用のシステムも増加しつつある。これらリアルタイムの処理システムでは、複数のタスクを同時・並行的に実行させることが多いため、各タスクのデッドラインが守られるようにスケジューリングすることが重要な課題になる。このようにタスク群をリアルタイム制約が守られるようにスケジューリングする技術を、リアル

タイムスケジューリングと呼んでいる。

【０００３】リアルタイムスケジューリングで多く用いられるスケジューリング方式は優先順位スケジューリングである。このスケジューリング方式は、タスクに優先順位を設定し、ＣＰＵを要求しているタスク中で最も優先順位の高いタスクにＣＰＵを利用させるものである。優先順位スケジューリングの中でもよく知られているのは、周期的に発生するタスクに対して周期が短いものほど優先順位を高く設定するレートモノトニックスケジューリングや、デッドラインまでの残り時間が短いものほど優先順位を高く設定するＥＤＦ(Earliet Deadline First)スケジューリングなどである（参考文献：C.L. Liu and J.W. Leyland, Scheduling algorithms for multi-programming in a hard real-time environment, in Journal of the Association for Computing Machinery, 20, 1 (January 1973):40—61）。

【０００４】これら優先順位スケジューリングは、優先順位が固定である固定優先順位スケジューリングと、優先順位が動的に変化する動的優先順位スケジューリングに大別される。例えば、レートモノトニックスケジューリングは固定優先順位スケジューリングの代表例であり、また、ＥＤＦスケジューリングは動的優先順位スケジューリングの代表例である。

【０００５】このようなリアルタイムスケジューリングにおいて、与えられたタスク群があるスケジューリング方式で、リアルタイム制約が充足されるようにスケジューリングできるかを調べたり、また、リアルタイム制約が守られるようにするにはどうスケジューリングすればよいかを調べる技術は、実用上重要である。このような技術を、スケジューリング解析技術という。

【０００６】スケジューリング解析技術の代表例としては、静的優先順位スケジューリングを主な対象とするＲＭＡ(Rate-Monotonic Analysis)を挙げることができ、これをもとにしたリアルタイムシステム構築手法も提唱されている（参考文献：M.H.Klein et al, A Practitioner's Handbook for Real-Time Analysis, KluwerAcademic Publ., 1993）。また、ＲＭＡに基づいたスケジューリング解析ツールもすでにいくつか存在する（例えば、J.W.S. Liu et al, PERTS: A prototyping environment for real-time systems, Proc. of 14th IEEE Real-Time Systems Symposium, 184—188, IEEE Press, 1994）。

【０００７】ＲＭＡに基づいたこれらスケジューリング解析に関する技術は、タスクＩＤ，優先順位、実行時間、周期、タスクスィッチに要する時間などの情報に基づき、タスク群がスケジューリング可能かどうかを検証し、その結果をユーザに提供するものである。

【０００８】

【発明が解決しようとする課題】ところで、上記のような従来のスケジューリング解析技術は、タスク毎の実行時間が常に一定であることを前提とし、リアルタイム制約が充足されるかどうかを二者択一で判断するものであった。一方、実際のタスクのプログラムは、ループや条件分岐を含むため、実行時間は必ずしも一定でなく、確率的に分布する。このため、従来のスケジューリング解析技術は、安全サイドに立って最悪の実行時間を想定することによってスケジューリング解析を行っていた。

【０００９】しかし、このようなスケジューリング解析では、最悪の実行時間が平均的な実行時間に比べて非常に大きいと、実行時間が最悪の場合に対してもスケジューリングできることを保証するためには、結果的に実行時間の余裕分が大きくなり、ＣＰＵが大半の場合に遊休状態となる。このようなスケジューリングのアプローチは、軍事・公共システムのように、リアルタイム制約の充足が最優先課題で、そのために多大のコストが許容されるシステムならば問題ない。

【００１０】しかし、コスト制約の厳しい民生品には、同様のアプローチは適当ではない。すなわち、民生品ではリアルタイム制約を絶対に保証したいという要求は小さく、十分高い確率でリアルタイム制約が充足されれば良いといった場合が多い。このため、従来技術を用いても民生品について適切なリアルタイムスケジューリングを行なうことは困難であった。

【００１１】また、従来の技術では、最悪の実行時間を前提としたが、例えばタスクのプログラム中にループが存在する場合は、ループの繰り返し回数に上限がなければ最悪の実行時間にも上限がないこととなる。このように、従来のスケジューリング解析技術は、ループや分岐を含むために実行時間が確率的に分布するようなタスク群に対しては適用が困難であり、適用範囲が狭いという問題点があった。

【００１２】また、組み込みシステムの設計においては、システムの機能をハードウェアとソフトウェアに適切に切り分けることによって、コストと性能との最適なトレードオフを実現するハード・ソフト協調設計技術(hardware/software codesign)が注目されている。このハード・ソフト協調設計技術においては、ある機能をハードウェアとソフトウェアのどちらで実現するかの選択が重要である。このような場合に対しても、スケジューリング解析は、ある機能をソフトウェアで実現した場合にデッドラインが破られるかどうか判定することによって、ハードウェアとソフトウェアの切り分けに関する判断材料を提供することができる。

【００１３】しかし、この場合も、これまでの手法は条件分岐やループを持つタスクの扱いが困難だったため適用範囲が狭く、適用できる場合でも、最悪の実行時間をもとにするためデッドラインが破られるという結論が不必要に多く、ある機能をハードウェアとソフトウェアのどちらで実現するかについて、妥当な判断は困難であった。

【００１４】本発明は、上記のような従来技術の問題点を解決するために提案されたものであり、その目的は、適用範囲が広く、タスクの実行時間の期待値に基づいて情報を提示するスケジューリング解析の技術を提供することである。より具体的には、本発明の目的は、実行時間が確率的な分布を持つタスク群を、指定されたスケジューリング方式でスケジューリングした場合に、リアルタイム制約が充足される可能性を算出して出力する技術を提供することである。なお、本出願人は、プログラムを入力とし、その実行時間の確率分布を出力する発明を出願しており（特願平０９－０４７６４３：酒井良哲／計算機システムの性能評価の方法及び装置）、この発明と本発明とを併せて利用することによって、複数のプログラムから、それらをスケジューリングした場合にリアルタイム制約が充足される可能性を直接・自動的に算出することができる。また、本発明の他の目的は、スケジューリングの可能性を安価に向上できる情報を提示する技術を提供することである。

【００１５】

【課題を解決するための手段】上記の目的を達成するため、請求項１のスケジューリング解析装置は、スケジューリングの対象とする各タスクに関する情報に基づいて、与えられたスケジューリング方式の下で、与えられたデッドラインまでに全てのタスクが実行を終了する可能性を算出する手段を備えたことを特徴とする。請求項５のスケジューリング解析方法は、請求項１の発明を方法の観点から把握したもので、スケジューリング方式を指定するステップと、実行すべき複数のタスクの実行周期又はデッドラインを指定するステップと、前記タスクの実行時間の確率的分布を用いて、指定されたスケジューリング方式において前記周期内又はデッドラインまでに全てのタスクが終了する可能性を示す値を算出するステップと、を含むことを特徴とする。請求項７の発明は、請求項１の発明をコンピュータのソフトウェアを記録した記録媒体の観点から把握したもので、コンピュータを用いて複数のタスクのリアルタイムスケジューリングに関する解析を行なうスケジューリング解析用ソフトウェアを記録した記録媒体において、当該スケジューリング解析用ソフトウェアはコンピュータに、スケジューリング方式の指定を受け付けさせ、実行すべき複数のタスクについて実行周期又はデッドラインの指定を受け付けさせ、前記タスクの実行時間の確率的分布を用いて、指定されたスケジューリング方式において前記周期内又はデッドラインまでに全てのタスクが終了する可能性を示す値を算出させることを特徴とする。請求項１，５，７の発明では、与えられたスケジューリング方式でタスク群をスケジューリングした場合について、デッドラインまでに実行が終了するかどうかを二者択一に判断するのではなく、各タスクの実行時間の期待値等に基づいて、終了する可能性がどのくらいかを算出して表示画面などに提示する。このため、ユーザは、リアルタイム制約が十分高い確率で満足されるかどうかをコストとの関係で具体的に判断することができる。

【００１６】請求項２の発明は、請求項１記載のスケジューリング解析装置において、前記算出する手段は、スケジューリングの対象とする各タスクごとに、多項式又は有理式として表された実行時間の確率分布関数、及び実行周期が与えられた場合に、有理式として与えられた確率分布関数を多項式に展開する手段と、各タスクの多項式に含まれる項の次数を、当該タスクの実行周期の逆数倍することによって、各タスク毎の第２の多項式を作成する手段と、各タスク毎の第２の多項式同士の積を計算することによって第３の多項式を作成する手段と、前記第３の多項式から、次数が、与えられたスケジューリング方式に応じて予め定めた基準値以下の項の係数の和を計算する手段と、前記係数の和に基づいて、前記スケジューリング方式の下で、与えられたデッドラインまでに全てのタスクが実行を終了する可能性を計算する手段と、を備えたことを特徴とする。請求項２の発明では、各タスクの実行時間の確率分布関数を、プログラムの内容に応じて多項式又は有理式で与えておくことにより、タスク毎及び全体としてのＣＰＵ使用率の期待値が計算され、スケジューリングの可能性が提示される。このため、実行時間が確率的に分布するタスク、例えばプログラム中に分岐やループを持つタスクについても適用対象とすることができる。

【００１７】請求項３の発明は、請求項１又は２記載のスケジューリング解析装置において、スケジューリングの対象とする各タスクに関する情報に基づいて、処理のデッドラインを破る原因となる可能性が高いタスクを同定する手段を備えたことを特徴とする。請求項６の発明は、請求項３の発明を方法の観点から把握したもので、請求項５記載のスケジューリング解析方法において、スケジューリングの対象とする各タスクに関する情報に基づいて、処理のデッドラインを破る原因となる可能性が高いタスクを同定するステップを含むことを特徴とする。請求項３，６の発明では、どのタスクがデッドラインを破る原因となる可能性が高いかが提示されるので、そのタスクを重点的にチューニングすることによって、新たなＣＰＵなどの高価な設備が不要となり、スケジューリングの可能性を安価に向上することができる。

【００１８】請求項４の発明は、請求項３記載のスケジューリング解析装置において、前記同定する手段は、スケジューリングの対象とする各タスクの実行時間の確率分布関数に基づいて、１回の実行あたりの実行時間について、期待値及びその分散を計算する手段と、１回の実行あたりの前記期待値及びその分散と当該タスクの実行周期とに基づいて、タスク毎のＣＰＵ使用率の期待値及びその分散を計算する手段と、タスク毎に計算されたＣＰＵ使用率の期待値又はその分散のうち少なくとも一方

に基づいて、デッドラインを破る原因となる可能性が高いタスクを判定する手段と、を備えたことを特徴とする。請求項4の発明では、デッドラインを破る原因となる可能性が高いタスクを、ＣＰＵ使用率の期待値やその分散から判定する。この場合、期待値からはＣＰＵ使用率の大きなタスクが同定され、分散からはＣＰＵ使用率の変動幅が大きいタスクが同定されるので、これらのタスクをチューニングすることによって、スケジューリング可能性を効果的に改善することができる。

【００１９】

【発明の実施の形態】次に、本発明の実施の形態であるスケジューリング解析装置（以下「本装置」という）について、図面を参照して具体的に説明する。なお、本発明は、周辺機器を持つコンピュータを、ソフトウェアで制御することによって実現することが一般的と考えられる。この場合、キーボード及びマウスなどの入力装置で情報を入力し、ＣＲＴ表示装置及びプリンタなどの出力装置で情報を出力できる。また、レジスタ、メモリ、外部記憶装置などの記憶装置は、いろいろな形式で、情報を一時的に保持したり永続的に保存できる。そして、ＣＰＵは、前記ソフトウェアにしたがって、これらの情報に加工及び判断などの処理を加え、さらに、処理の順序を制御することができる。

【００２０】また、コンピュータを制御するソフトウェアは、各請求項及び明細書に記述する処理に対応した命令を組み合わせることによって作成され、作成されたソフトウェアは、コンパイラやインタプリタなどの処理系によって実行されることで、上記のようなハードウェア資源を活用する。

【００２１】但し、本発明を実現するための上記のような態様はいろいろ変更することができ、例えば、本発明の装置と外部との間で情報を入出力するには、フロッピーディスクなどの着脱可能な記録媒体やネットワーク接続装置を使用することもできる。さらに、本発明を実現するソフトウェアを記録したＣＤ－ＲＯＭのような記録媒体は、それ単独でも、本発明の一態様である。また、本発明の機能の一部をＬＳＩなどの物理的な電子回路で実現することも可能である。

【００２２】以上のように、コンピュータを使用して本発明を実現する態様はいろいろ変更できるので、以下では、本発明の各機能を実現する仮想的回路ブロックを用いることによって、本発明の実施の形態を説明する。

【００２３】〔１．構成〕まず、図１は、本装置の構成を示す機能ブロック図である。すなわち、本装置は、この図に示すように、タスク群に属する各タスクに関するタスク情報を保持するタスク情報記憶部１０と、各タスクの情報からスケジューリング方式のもとでリアルタイム制約が充足される可能性を算出するスケジューリング可能性算出部１１と、デッドラインを破る原因となる可能性が高いタスク（以下「原因タスク」と呼ぶ）を指摘するタスク同定部１２と、を有する。ここで、与えられたスケジューリング方式の下でリアルタイム制約が充足されること、すなわち、全てのタスクがデッドラインまでに実行を終了する確率を、スケジューリング可能性と呼ぶ。

【００２４】また、本装置は、スケジューリング可能性算出の前提とするスケジューリング方式を指定するためのスケジューリング方式指定部１４と、スケジューリング可能性算出部１１及びタスク同定部１２の出力結果をもとに画面表示を行う結果表示部１３とを有する。

【００２５】さらに、スケジューリング可能性算出部１１は、展開部１１１と、項の次数計算部１１２と、積の計算部１１３と、項の次数除外部１１４と、項の係数総和部１１５と、可能性計算部１１５と、を有する。ここで、スケジューリング可能性算出部１１には、スケジューリングの対象とする各タスクごとに、多項式又は有理式として表された実行時間の確率分布関数、及び実行周期が与えられるが、展開部１１１は、与えられた確率分布関数が有理式であった場合に、多項式に展開する手段である。また、項の次数計算部１１２は、各タスクの多項式に含まれる項の次数を、当該タスクの実行周期の逆数倍することによって、各タスク毎の第２の多項式を作成する手段である。

【００２６】また、積の計算部１１３は、各タスク毎の第２の多項式同士の積を計算することによって第３の多項式を作成する手段である。また、項の係数総和部１１４は、前記第３の多項式から、次数が、与えられたスケジューリング方式に応じて予め定めた基準値以下の項の係数の和を計算する手段である。また、可能性計算部１１６は、前記係数の和に基づいて、前記スケジューリング方式の下で、与えられたデッドラインまでに全てのタスクが実行を終了する可能性を計算する手段である。

【００２７】また、タスク同定部１２は、さらに、第１の期待値・分散計算部１２１と、第２の期待値・分散計算部１２２と、判定部１２３と、を有する。このうち、第１の期待値・分散計算部１２１は、スケジューリングの対象とする各タスクの実行時間の確率分布関数に基づいて、１回の実行あたりの実行時間について、期待値及びその分散を計算する手段である。

【００２８】また、第２の期待値・分散計算部１２２は、１回の実行あたりの前記期待値及びその分散と当該タスクの実行周期とに基づいて、タスク毎のＣＰＵ使用率の期待値及びその分散を計算する手段である。また、判定部１２３は、タスク毎に計算されたＣＰＵ使用率の期待値又はその分散のうち少なくとも一方に基づいて、デッドラインを破る原因となる可能性が高いタスクを判定する手段である。

【００２９】〔２．作用〕上記のような構成を有する本実施形態では、次のような処理によって、スケジューリング可能性及び原因タスクがユーザに提示される。な

お、スケジューリングの対象とする各タスク毎に、あらかじめユーザが、確率分布関数である確率母関数ｆ及び実行周期Ｔをタスク情報記憶部１０に入力しておき、また、スケジューリングに用いようとするスケジューリング方式をスケジューリング方式指定部１４から指定しておくものとする。ここで、母関数とは、主に数え上げに用いられ、漸化式などの記述を簡単にするもので、確率母関数とは、ありうる複数の事象それぞれが発生する確率を記述した母関数である。

【００３０】〔２－１．スケジューリング可能性を算出する理論的根拠〕本実施形態では、スケジューリング可能性算出部１１が、スケジューリングの可能性を計算するが、このような計算の理論的根拠をまず説明する。ここで、図２は、タスク情報記憶部１０に保持されているタスク情報の例を示す図である。このタスク情報は、タスク毎に、タスクＩＤ、周期、実行時間の確率分布を含む。このうちタスクＩＤは、タスク毎に一意につけられる固有の識別子である。また、そのタスクが周期的に起動されるものの場合は、周期を記録する。なお、この周期の欄は、非周期的なタスクの場合は空欄とする。

【００３１】実行時間の確率分布は、確率母関数またはｚ－変換を用いれば、分岐・ループがある場合も、有理式としてコンパクトに表現できる。ここでは簡単なため、すべての時間はある単位時間の何倍という形の無次元量として扱う。すなわち、図２に示した確率分布は、有理式を用いた確率分布を表現する例を示したもので、この場合、多項式は、分母が１の有理式として扱う。

【００３２】この例において、確率分布が有理式Ｐ／Ｑ（Ｐ，Ｑは多項式）である場合、

【数１】$P=\Sigma_s\ a_s w^s\ (a_s \neq 0)$

を、ペア（ａｓ　ｓ）からなるリストＬ1 で表し、

【数２】$Q=\Sigma_t\ b_t w^t\ (b_t \neq 0)$

を、ペア（ｂｔ　ｔ）からなるリストＬ2 で表すことによって、有理式Ｐ／Ｑにを、これらリストのペア（Ｌ1　Ｌ2）で表す。なお、本明細書において、ｘのｙ乗は「x^y」の他、「ｘ＾ｙ」及び「ｘ＊＊ｙ」のように表し、また、以上（≧）や以下（≦）は「>=」「=<」のようにも表す。

【００３３】以下では、タスクｔ１，ｔ2 …ｔｎ は周期的・独立と仮定し、その周期をＴ1，Ｔ2 …Ｔｎ とする。タスクｔｉ の実行時間Ｃｉ は確率的な分布に従うが、ここでは実行時間Ｃｉ は実行される全てのインスタンスで一定と仮定する。「インスタンス」は、同じプログラムに基づいた一連の処理が繰り返し行なわれる場合において、個々の一連の処理である。

【００３４】このように同種のタスクのインスタンス間で実行時間Ｃｉ が一定という仮定が妥当するのは、プログラムの文面からは確率的にしか実行時間が見積もれないが、実際の実行では実行パスやループの回数が一定となる場合であり、組み込みソフトに多く見られる。このようなタスクのインスタンスは、そのタスクの次のインスタンスが発生する時点をデッドラインとして、それまでに実行時間分だけＣＰＵを獲得する必要がある。そして、すべてのタスクインスタンスが、デッドライン以内に実行時間分ＣＰＵを獲得できる時、タスク群はスケジューリング可能であるという。この仮定は、前掲のLiu-Layland の論文においても用いられていた。

【００３５】ここで、Ｘを、実数に値を持つ離散的ランダム変数とする。このような離散的ランダム変数Ｘの確率母関数ｆＸ（ｗ）は、次のように定義する：

【数３】$fX(w)=\Sigma_r\ Pr(X=r)\ w^r$

ここで、Ｐｒ（Ｘ＝ｒ）は、事象Ｘ＝ｒの確率である。上の例のように、変数Ｘが整数にのみ値をもつランダム変数である場合は、確率母関数ｆＸ（ｗ）は信号処理で言うところの信号のｚ－変換などと大体同じものであるが、ここではＸが整数値にかぎらず実数値を取りうる点に特徴がある。

【００３６】ここで、ランダム変数Ｃｉ の確率母関数

【数４】$fCi(w)=\Sigma_r\ Pr(Ci=r)\ w^r$

を考える。この場合、ｆＣｉ（ｗ）は無限に多くの項を持つ可能性があるが、実際のプログラムで用いられる条件分岐やループの影響を考えると、ｗの多項式または有理式となる。なお、プログラムに含まれる単なる実行文以外の要素が条件分岐だけならば多項式であるが、プログラム中にループが存在しても有理式として扱うことができる。特に、ｆＣｉ（ｗ）の情報は有限のメモリで保持できる。

【００３７】続いて、実行時間の確率母関数の例を二つ与える。但し、ここでの説明は簡単化したものであり、必ずしも実際に用いられる例に即したものではない。まず、プログラムのある断片として、簡単な分岐を持つプログラム

```
if  e1 then  e2 elsee3
```

の実行時間を考える。この例では、条件判定部ｅ１にかかる時間が無視でき、各実行文ｅ２，ｅ３それぞれへの分岐確率は共に５０％、実行文ｅ２の実行に要する時間は４、実行文ｅ３の実行に要する時間は８であるとする。この場合、

```
if  e1 then  e2 else  e3
```

の実行時間の確率母関数は、

【数５】$1/2\ w^4+1/2\ w^8$

で与えられる。

【００３８】次に、プログラムの別の断片として、簡単なループを持つプログラム

```
e4 ; while e5 do  e6 ;
```

を考える。この例では、実行文ｅ４の実行時間を５、条件判定部ｅ５にかかる時間が無視でき、そこで条件が成立する確率は５０％、実行文ｅ６の実行に要する時間を４とする。この実行時間の確率母関数は、

【数６】

$1/2\ w^4 + 1/4\ w^4 + 1/8\ w^{12} + \cdots$
$= 1/2\ w^4\ (1 - 1/2\ w^4)^{-1}$
となる。このような例では、ループが回る回数には上限がないが、確率母関数は有理式であり、有限のメモリや図表などの形式で保持できる。この事実は、非常に広い範囲のプログラムに対しても成立することが知られているので、以下、この前提で説明を続ける。

【００３９】なお、確率母関数は、上記に例示したように、プログラムのソースコードの内容に応じて定まるので、ユーザがタスクのプログラム毎に手作業で作成して入力しておいてもよいが、ソースコードの内容をプログラミング言語の文法に基づいて解析することによって、自動的に生成することもできる。

【００４０】そして、ＣＰＵ使用率Ｕは、次の式で与えられるランダム変数である：

【数７】$U = \Sigma_{i=1}^{n} (Ci / Ti)$

また、タスクｔｉ単独のＣＰＵ使用率Ｕｉを

【数８】$Ui = Ci / Ti$

として定義する。この場合、タスクｔｉ単独のＣＰＵ使用率Ｕｉの確率母関数ｆＵｉ（ｗ）を考えると、

【数９】
$fUi(w) = \Sigma_r Pr(Ci/Ti = r)\, w^r$
$= \Sigma_r Pr(Ci = Ti \cdot r)\, w^r$
$= \Sigma_s Pr(Ci = s)\, w^{s/Ti}$
$= \Sigma_s Pr(Ci = s)\, (w^{1/Ti})^s$
$= fCi(w^{1/Ti})$

【数１０】$U = \Sigma_{i=1}^{n} Ui$

であるが、全てのタスクが互いに独立との仮定から、

【数１１】$fU(w) = \Pi_{i=1}^{n} fUi(w)$

となる。上の等式から、

【数１２】$fU(w) = \Pi_{i=1}^{n} fCi(w^{1/Ti})$

となり、全タスクのＣＰＵ使用率の合計ｆＵ（ｗ）がタスクｔｉの１回当たりの実行時間ｆＣｉ（ｗ）と実行周期Ｔｉによって記述できることが確認できた。

【００４１】さらに、級数

【数１３】$f(w) = \Sigma_s as\, w^s$

のｒ次の係数a^rを、ｆ（ｗ）［w^r］で表わす。また、ｆ（ｗ）に現れる次数ｒ以下の係数の和$\Sigma_{s=<r} as$を、ｆ（ｗ）［$w^{=<r}$］で表わし、加えて、次のような類似の記法を導入しておく。

【数１４】
$f(w)\ [w^{=r}] = \Sigma_{s=r} as$
$f(w)\ [w^{>=r}] = \Sigma_{s>=r} as$
$f(w)\ [w^{>r}] = \Sigma_{s>r} as$

ここで、ランダム変数Ｘの確率母関数ｆＸ（ｗ）については、定義から次が成立する。

【数１５】
$Pr(X = r) = fX(w)\ [w^r]$
$Pr(X \leqq r) = fX(w)\ [w^{=<r}]$

また、二つの級数

【数１６】
$f(w) = \Sigma_s as\, w^s$, $g(w) = \Sigma_s bs\, w^s$

が、ある実数ｒより高次の項を無視して一致することを、

【数１７】$f(w) \equiv g(w)\ (\mathrm{mod}.\ w^{>r})$

で表わす。

【００４２】そして、前掲のLiu-Layland の論文では、周期的なタスクの集合がＥＤＦでスケジューリング可能であるための必要十分条件は、Ｕが１以下であることが示されている。このことから、あるタスク群がＥＤＦでスケジューリング可能である確率は

【数１８】$Pr(U \leqq 1) = fU(w)\ [w^{=<1}]$

となり、

【数１９】$(\Pi_{i=1}^{n} fCi(w^{1/Ti}))\ [w^{=<1}]$

を計算することで求められる。

【００４３】レートモノトニック方式の場合には、同じくLiu-Layland の論文で、ｎ個の周期的なタスクがレートモノトニック方式でスケジューリング可能であるための十分条件が、

【数２０】$U \leqq n(2^{1/n} - 1)$

であることが示されている。スケジューリング可能であるには、Ｕ≦１が必要であるから、

【数２１】
$Pr(U \leqq n(2^{1/n} - 1))$
$\leqq Pr(\text{レートモノトニックスケジューリング可能})$
$\leqq Pr(U \leqq 1)$

が成立する。また、

【数２２】$Pr(U \leqq 1) = fU(w)\ [w^{=<1}]$
$Pr(U \leqq n(2^{1/n} - 1)) = fU(w)\ [w^{=<n(2^{**}(1/n)-1)}]$

を使って、レートモノトニック方式でスケジューリング可能な確率の上界と下界が計算できるが、何%であるといった正確な主張はできない。以上が、本実施の形態において、スケジューリング可能性を算出する理論的な根拠である。

【００４４】〔２－２．スケジューリング可能性を算出する手順〕続いて、タスク群の情報とタスクスケジューリング方式からスケジューリング可能性を算出する際の処理手順を、図３のフローチャートに基づいて説明する。なお、同じタスクの組み合わせでも、どのスケジューリング方式でスケジューリングするかによってスケジューリング可能性は異なるので、判定の基準とするスケジューリング方式は、スケジューリング方式指定部１４からユーザが予め指定しておく。また、この際、指定されたスケジューリング毎に予め決まっている基準値が選択される。この基準値は、スケジューリング可能性を算出する際に用いるものである。

【００４５】図３のフローチャートに示す処理手順では、まず、スケジューリング可能性算出部１１が、タスク情報記憶部１０からタスク情報として各タスクの確率

母関数ｆと実行周期Ｔを読み出す（ステップ３１）。

【００４６】ここで、各タスクｔｉの実行時間Ｃｉの確率分布関数ｆＣｉ（ｗ）は多項式または有理式の形で与えられていると仮定するが、展開部１１１は、有理式として与えられたものはここで多項式に展開する（ステップ３２）。このように有理式を展開する場合、そのままでは多項式の形式にならず、無限に項が続く場合があるが、項の次数が周期Ｔｉより長いものは後の計算で利用しないため無視できるので、多項式とみなしてよい。このような多項式は係数と次数のペアのリストとして、有理式は多項式のペアとして表現される。この結果、全てのタスクについて、実行時間Ｃｉの確率分布を表す多項式が用意される。

【００４７】続いて、項の次数計算部１１２が、各タスクの確率分布関数ｆＣｉ（ｗ）に現れる各項の次数を、当該タスクの実行周期の逆数倍、すなわち１／Ｔｉ倍することによって第２の多項式を作成する（ステップ３３）。この処理は、タスク毎にＣＰＵ使用率を計算することを意味する。

【００４８】そして、積の計算部１１３が、前記第２の多項式同士を掛け合わせた積を計算することによって第３の多項式を作成する（ステップ３４）。この処理は、タスク毎に計算したＣＰＵ使用率から、スケジューリングの対象となるタスク群全体について、ＣＰＵ使用率の合計を計算することを意味する。

【００４９】ここで、予め指定されているスケジューリング方式に応じた基準値が選択されており、項の係数総和部１１４が、前記第３の多項式に現れる各項のうち、＊

$$fU1(w)=fC1(w^{1/15})=1/2\,w^{4/15}+1/2\,w^{8/15}$$

$$fU2(w)=fC2(w^{1/20})=1/2\,w^{1/5}\,(1-1/2\,w^{1/15})^{-1}$$

$$\equiv 1/2\,w^{1/5}+1/4\,w^{2/5}+1/8\,w^{3/5}+1/16\,w^{4/5}+1/32\,w\ (\mathrm{mod}.\ w^{>1})$$

となる（第２の多項式）。続くステップ３４において、項の次数が逆数倍された各多項式同士の積をとり（第３の多項式）、この積から、次数が１より大きな項は廃棄※

$$fU(w)\equiv(1/2\,w^{4/15}+1/2\,w^{8/15})\cdot(1/2\,w^{1/5}+1/4\,w^{2/5}+1/8\,w^{3/5}+1/16\,w^{4/5}+1/32\,w)\ (\mathrm{mod}.\ w^{>1})$$

$$\equiv 1/4\,w^{4/15}+1/8\,w^{7/15}+1/16\,w^{10/15}+1/4\,w^{8/15}+1/8\,w^{11/15}+1/16\,w^{14/15}\ (\mathrm{mod}.\ w^{>1})$$

となる。また、ステップ３５では、スケジューリング方式がＥＤＦの場合、基準値は１であり、次数が１以下の項の係数の和を

【数２７】

$$Pr(U\leqq 1)=fU(w)[w^{-<1}]$$

$$=1/4+1/8+1/16+1/4+1/8+1/16$$

$$=7/8=0.875$$

のように計算する。同様に、スケジューリング方式がレートモノトニックの場合は、次数が２（$2^{1/2}-1$）＝０．８２８４以下の項の係数の和も

＊次数が基準値より大きい項は廃棄し（ステップ３４）、次数が基準値以下の項のみを選んで項の係数の総和を計算する（ステップ３５）。最後に、可能性計算部１１５が、計算された係数の総和に基づいてスケジューリング可能性を計算して結果表示部１３に出力する（ステップ３６）。

【００５０】〔２－３．スケジューリング可能性を算出する実例〕次に、具体的な数値を用いて、スケジューリング可能性を算出する例を示す。例えば、タスクｔ１，ｔ２として、既に上で与えた分岐およびループの例を使うと、実行時間Ｃ１，Ｃ２の確率分布関数は、

【数２３】

$$fC1(w)=1/2\,w^{4}+1/2\,w^{8}$$

$$fC2(w)=1/2\,w^{4}\,(1-1/2\,w^{4})^{-1}$$

となる。また、タスクｔ１の周期が１５、ｔ２の周期が２０であったとする。

【００５１】この場合、図３に示したステップ３１では、タスク情報記憶部１０からタスク情報を読み出し、続くステップ３２では、読み出したタスク情報のうち、確率母関数ｆＣｉ（ｗ）を多項式の形に展開する。この展開の結果をｆＣ２（ｗ）について例示すると、

【数２４】

$$fC2(w)\equiv 1/2\,w^{4}+1/4\,w^{8}+1/8\,w^{12}+1/16\,w^{16}+1/32\,w^{20}\ (\mathrm{mod}.\ w^{>20})$$

となる。そして、ステップ３３では、確率母関数ｆＣｉ（ｗ）に現れる項の次数を１／Ｔｉ倍する。これを実行すると、

【数２５】

※する。これによって、

【数２６】

【数２８】

$$Pr(U\leqq 2(2^{1/2}-1))=fU(w)[w^{-<2(2^{*}(1/2)-1)}]$$

$$=1/4+1/8+1/16+1/4+1/8$$

$$=13/16=0.8125$$

のように、計算する。最終的に、ステップ３６において、ＣＰＵ使用率とスケジューリング可能性の関係から、

【数２９】Ｐｒ（ＥＤＦ方式でスケジューリング可能）＝０．８７５

15

0．8125≦Pr（レートモノトニック方式でスケジューリング可能）≦0．875

なので、「EDF方式でスケジューリング可能な確率は87．5%」・「レートモノトニック方式でスケジューリング可能な確率は81．25%以上87．5%以下」などを表示する。

【0052】〔2－4．原因タスクを同定する手順〕ユーザは、上記のように得られたスケジューリング可能性の結果を見て、「スケジューリング可能な確率がこの程度では不安だ、何とかしてスケジューリング可能性をもっと高くしたい」と考えたとする。このようにスケジューリング可能性を高くするために、もっとも簡単・確実な方法は、CPUを能力が高いものに変更することである。このため、CPUの能力をどの程度向上させれば、スケジューリング可能性が十分高くできるかを調べることは、本発明の重要な用途である。例えば、CPUの処理能力を25%向上させると、タスクの実行時間は、向上させる前と比較してほぼ80%に短縮されると考えられる。但し、この実行時間の改善と、スケジューリング可能性の向上の間には簡単な関係は期待できないため、CPUの処理能力を向上させた場合の正確なスケジューリング可能性は、上に示した手順にしたがって再計算する必要がある。

【0053】しかし、このようにスケジューリング可能性をより安価に改善するためには、単にタスク群がリアルタイム制約を充足するようスケジューリングできる確率を提示するだけでなく、次のような情報を提供することが望ましい。すなわち、上記のようにCPUをより高速なものに取り替えることはスケジューリング可能性を向上させるための最も確実な解決策であるが、コストが増大するという問題がある。

【0054】これに対して、民生品のようにコストを強く意識する必要がある場合は、ユーザはシステムをチューニングすることだけですませ、高価なハードウェアを使用するのは可能な限り避けることが必要となる。しかし、全タスクをチューニングするのは、ユーザにかかる負担が大きい。ここで、どのタスクがデッドラインを破る原因になる可能性が高いかを判定してユーザに提示できれば、そのタスクを念入りにチューニングすることでスケジューリング可能性を向上できる確率が高く、ユーザに無駄な負担をかけることがない。そして、どのタスクがデッドラインを破る原因と思われるかを判定するには、タスク毎に定義できる何らかの指標が必要である。

【0055】ここで、これまでみてきたように、CPU使用率はスケジューリング可能性に大きく影響する。そのため、それぞれのタスクtiに関するCPU使用率Uiの期待値E［Ui］をこの指標として用いることが自然である。また、期待値E［Ui］以外の重要な指標としては、分散V［Ui］がある。すなわち、ランダム変数Xの分散V［X］は、

16

【数30】$V[X]=E[X^2]-(E[X])^2$

で定義される量である。

【0056】この分散は、次のような意味を持つ。すなわち、二つのタスクのCPU使用率が同じでも、一方は比較的期待値の近辺に集中して分布し、他方は大きなばらつきをもつような場合が考えられる。この場合、スケジューリング可能性の立場からは、後者のタスクの方がデッドラインを破る原因となる可能性が大きい。この状況は、前者の分散は後者の分散より大きいという形で特徴づけられる。

【0057】以下では、タスク毎のCPU使用率の期待値および分散を計算し、それらに基づいて原因タスクを同定する際の処理手順を、図4のフローチャートに基づいて説明する。すなわち、この手順では、まず、タスク同定部12が、タスク情報記憶部10からタスク情報を読み出す（ステップ41）。続いて、第1の期待値・分散計算部121が、確率母関数から、1回の実行あたりの実行時間Ciの期待値・分散をそれぞれ計算する（ステップ42）。ここで、確率母関数が多項式の場合、1回の実行あたりの期待値E［Ci］・分散V［Ci］は直接計算できる。一方、確率母関数が有理式の場合は、微分等を行って計算するが、詳細は後に実例の説明で述べる。

【0058】続いて、このように計算された1回の実行あたりの期待値E［Ci］・分散V［Ci］と、周期Tiとに基づいて、第2の期待値・分散計算部122が、タスク毎の

【数31】

期待値$E[Ui]=E[Ci]/Ti$

分散$V[Ui]=V[Ci]/Ti^2$

を計算する（ステップ43）。そして、判定部123が、CPU使用率の期待値・分散それぞれについて、大きい順にタスクをソートしたうえ（ステップ44）、CPU使用率の期待値・分散のどちらかもしくは両方がタスク全体で上位に位置するタスクを原因タスクとして同定し、結果表示部13に表示する（ステップ45）。

【0059】〔2－5．原因タスクを同定する実例〕例えば、周期的タスクの場合、タスク毎の期待値E［Ui］と分散V［Ui］を求めるには、まず、図4のステップ42において、1回の実行あたりのE［Ci］・V［Ci］を求めて、

【数32】

$E[Ui]=E[Ci]/Ti$

$V[Ui]=V[Ci]/Ti^2$

により、ステップ43において、E［Ui］・V［Ui］を計算するのが賢明である。ここでは、スケジューリング可能性を計算する実例と同じ例について、E［Ui］を求める処理の流れを説明する。すなわち、スケジューリング可能性の計算で示した例では、タスクの実行時間の確率母関数は、次の通りであった：

【数３３】
$fC1(w)=1/2\ w^4+1/2\ w^8$
$fC2(w)=1/2\ w^4\ (1-1/2\ w^4)^{-1}$
ここで、
【数３４】$fX(w)=\Sigma_i\ a_i\ w^i$
が有限和である場合はXの取る値が有限個であり、期待値E［X］もΣi ai sとして直接計算できる。期待値E［C1］は、このように直接計算できる例である。すなわち、
【数３５】
$E[C1]=1/2\cdot 4+1/2\cdot 8=6$
$U1=C1/T1=C1/15$
から、
【数３６】$E[U1]=E[C1]/15=2/5$
が求まる。タスクt2については、確率母関数が多項式ではないので、次の公式を用いる。すなわち、Xが離散的な場合には確率母関数fX（w）により、
【数３７】
$E[X]=(fX(w))'|_{w=1}$
$fC2(w)'=-2w^3\ (1-1/2\ w^4)^{-1}+w^7\ (1-1/2\ w^4)^{-2}+2w^3\ (1-1/2\ w^4)^{-1}+w^7\ (1-1/2\ w^4)^{-2}$
から、
【数３８】
$E[C2]=(fC2(w))'|_{w=1}=-2\times 2+22+2\times 2+4=8$
であり、
【数３９】
$E[U2]=E[C2]/20=8/20=2/5$
として計算できる。この計算結果から判断しようとすると、二つの平均値が一致するので、この指標についてはどちらのタスクが問題が多いとも言えない。
【００６０】ここで、
【数４０】$fX(w)=\Sigma_i\ a_i\ w^i$
が有限和である場合は、分散V［X］も直接計算できる。ここでも、V［U1］を直接計算するのではなく、V［C1］から関係
【数４１】$V[U1]=V[C1]/T1^2$
を用いて求める。すなわち、
【数４２】
$E[C1^2]=1/2\times 4^2+1/2\times 8^2=40$
であるから、
【数４３】$V[C1]=E[C1^2]-E[C1]^2=40-6^2=4$
さらに、
【数４４】
$V[U1]=V[C1]/15^2=4/225$
タスクt2については、Xが離散的な場合には確率母関数fX（w）により、
【数４５】$V[X]=(fX(w))''|_{w=1}+(fX(w))'|_{w=1}-((fX(w))'|_{w=1})^2$
として計算できることを用いる。つまり、
【数４６】
$fC2(w)''=6w^2\ (1-1/2\ w^4)^{-1}+11w^6\ (1-1/2\ w^4)^{-2}+4w^{10}(1-1/2\ w^4)^{-3}$
$(fC2(w))''|_{w=1}=6\times 2+11\times 2^2+4\times 2^3=88$
となるので、
【数４７】$V[C2]=88+8-8^2=32$
であり、このことから、
【数４８】$V[U2]=V[C2]/20^2=32/400=2/25$
である。
【００６１】以上の結果から、t1とt2のＣＰＵ使用率の期待値は同じであるが、分散はt2の方が大きいことがわかる。これ自体は、t1と異なりt2はループを含んでいることから自然である。
【００６２】なお、デッドラインを破る原因となる可能性の高いタスクを同定するための基準としては、上記の期待値と分散以外にも、いくつかの指標が考えられることは言うまでもない。しかし、期待値と分散は工学において広く用いられているもので、多くの状況に適用できると考えられる。この二つの指標は、独立に使用することもできるが、両者を併用することも可能である。その場合は、まず期待値を比較し問題のあるタスクを指摘しそれで検出できなかった問題点を分散を指標としてチェックする、といったアプローチが考えられる。具体的には、上の例で言うと、t2の方がデッドラインが破られる原因となる可能性が高いことが結論できる。
【００６３】〔３．効果〕以上説明したように、本実施の形態によれば、最悪の実行時間ではなく、典型的な場合の実行時間を前提にスケジューリング可能性を算出するので、スケジューリング解析技術の適用可能範囲を広げ、スケジューリング可能性に関するより現実的な指針を提供することができる。すなわち、本実施の形態では、与えられたスケジューリング方式でタスク群をスケジューリングした場合について、デッドラインまでに実行が終了するかどうかを二者択一に判断するのではなく、各タスクの実行時間の期待値等に基づいて、終了する可能性がどのくらいかを算出して表示画面などに提示する。このため、ユーザは、リアルタイム制約が十分高い確率で満足されるかどうかをコストとの関係で具体的に判断することができる。また、このような本実施の形態を応用することによって、ハード・ソフト協調設計支援技術の実用性が大幅に向上する。
【００６４】また、本実施の形態では、各タスクの実行時間の確率分布関数を、プログラムの内容に応じて多項式又は有理式で与えておくことにより、タスク毎及び全体としてのＣＰＵ使用率の期待値が計算され、スケジュ

ーリングの可能性が提示される。このため、実行時間が確率的に分布するタスク、例えばプログラム中に分岐やループを持つタスクについても適用対象とすることができる。

【００６５】また、本実施の形態では、どのタスクがデッドラインを破る原因となる可能性が高いかが提示されるので、そのタスクを重点的にチューニングすることによって、新たなＣＰＵなどの高価な設備が不要となり、スケジューリングの可能性を安価に向上することができる。

【００６６】また、本実施の形態では、デッドラインを破る原因となる可能性が高いタスクを、ＣＰＵ使用率の期待値やその分散から判定する。この場合、期待値からはＣＰＵ使用率の大きいタスクが同定され、分散からはＣＰＵ使用率の変動幅が大きいタスクが同定されるので、これらのタスクをチューニングすることによって、スケジューリング可能性を効果的に改善することができる。

【００６７】〔４．他の実施の形態〕なお、本発明は上記実施の形態に限定されるものではなく、次に例示するような他の実施の形態も包含するものである。例えば、スケジューリング可能性算出の前提となるスケジューリング方式としては、上記実施の形態に示したもののほか、所望のものを用いることができる。また、実行時間の確率分布関数は実質的に多項式又は有理式として表されていればよく、内部構造としては数値のペアなど所望のデータ形式で保持することができる。

【００６８】

【発明の効果】以上説明したように、本発明によれば、適用範囲が広く、タスクの実行時間の期待値に基づいて＊情報を提示するスケジューリング解析の技術を提供できるので、民生品についてもより安価にリアルタイムスケジューリングを適用することが可能となる。

【図面の簡単な説明】

【図１】本発明の実施の形態の構成を示す機能ブロック図。

【図２】本発明の実施の形態において、タスク情報記憶部１０に保持されているタスク情報を例示する図。

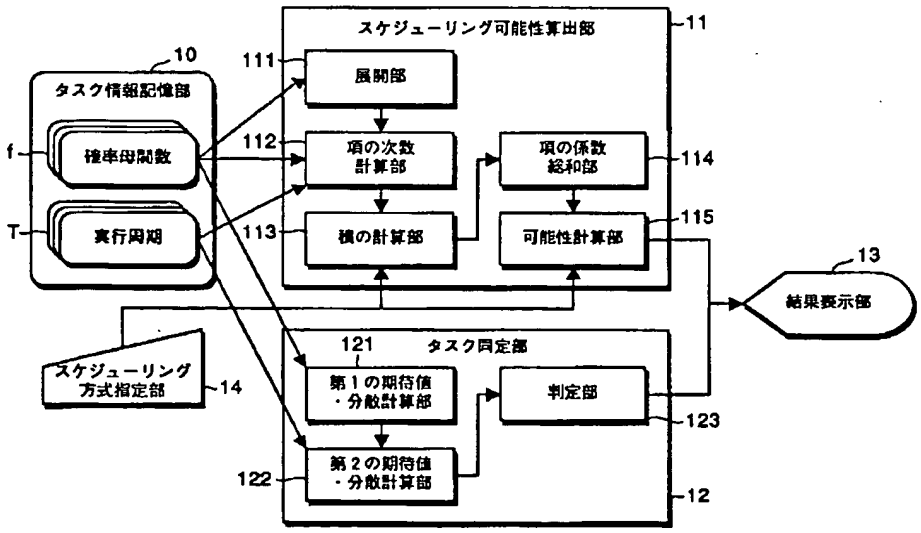
【図３】本発明の実施の形態において、タスク群の情報及びスケジューリング方式からスケジューリング可能性を算出する際の処理手順を表わすフローチャート。

【図４】本発明の実施の形態において、タスク群の情報及びスケジューリング方式から、タスク群がデッドラインを破る原因となる可能性が高いタスクを同定する際の処理手順を表わすフローチャート。

【符号の説明】

１０…タスク情報記憶部
１１…スケジューリング可能性算出部
１１１…展開部
１１２…項の次数計算部
１１３…積の計算部
１１４…項の係数総和部
１１５…可能性算出部
１２…タスク同定部
１２１…第１の期待値・分散計算部
１２２…第２の期待値・分散計算部
１２３…判定部
１３…結果表示部
１４…スケジューリング方式指定部

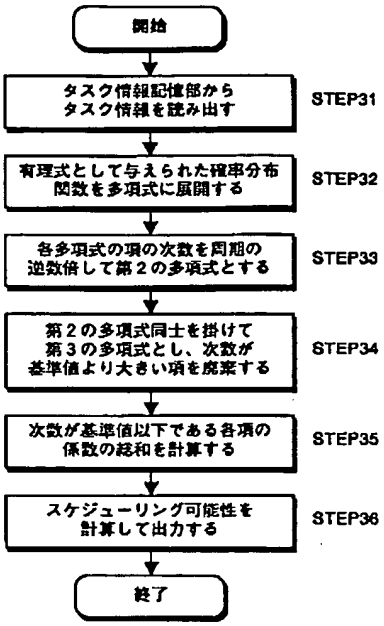
【図１】



【図２】

タスク1: 周期15; 実行時間の分布確率 (((0.5 5) (0.5 10)) ((1 0)　　))
タスク2: 周期20; 実行時間の分布確率 (((0.5 5)　　) ((1 0) (-0.5 5)))

【図３】



【図４】